Л.Деткова,
учитель информатики первой квалификационной категории

И.Сухоручкина,
к. т. н., педагог дополнительного образования высшей квалификационной категории, руководитель Консультативного центра

М.Севастьянов, обучающийся 10 класса,
ГБОУ «Одинцовский «Десятый лицей», Московская область г. Одинцово

ЦИФРОВОЙ ГОЛОСОВОЙ ПОМОЩНИК: ПРОЕКТ ПО ИНФОРМАТИКЕ И ПРОГРАММИРОВАНИЮ

ГОЛОСОВЫЕ помощники используются в разных видах деятельности, бизнеса и обеспечивают высокоэффективную интерактивную коммуникацию. Будущим программистам мы предлагаем наш опыт создания голосового ассистента. Цель проекта – создание голосового ассистента на основе языка программирования Python. Задачи проекта: 1) изучение основных концепций создания голосовых ассистентов, функций и программных решений; 2) установка на компьютер программ и получение базовых знаний; 3) написание базовой программы для ассистента и его навыков.

Методы проекта. Для написания программы голосового ассистента использованы дополнительные библиотеки языка программирования Python: PyAudio. URL: https://github.com/jleb/pyaudio; Wikipedia API. URL: https://github.com/martin-majlis/Wikipedia-API; google 3.0.0. URL:https://pypi.org/project/google/; pyttsx3. URL: https://github.com/nateshmbhat/pyttsx3; SpeechRecognition. – URL: https://github.com/Uberi/speech_recognition; deep-translator. URL: https://github.com/nidhaloff/deep-translator.

Эти библиотеки полностью свободны, с открытым исходным кодом.

Наш проект также будет свободным. Весь код доступен в репозиториях GitHub. Ссылка GitHub проекта: https://github.com/matvey100/victoria-assistant. Желающие могут скачать и собрать его.

Это один из главных принципов создания, в отличие от коммерческих ассистентов Яндекс Алиса (URL: https://yandex.ru/alice), Google Assistant (URL: https://assistant.google.com/intl/ru_ru/) и Apple Siri (URL: https://www.apple.com/ru/siri/).

Его достоинство – кроссплатформенность, возможность выполнения кода на разных операционных системах и платформах Linux (URL: https://www. linux.org/), Windows (URL: https://www.microsoft.com/en-us/windows?r=1), MacOS (URL: https://support.apple.com/ru-ru/HT201260), Android (URL: https:// www.android.com/intl/ru_ru/) и iOS (URL: https://www.apple.com/ios/ios-16/).

Ассистент может говорить разными голосами и на разных языках.

Реализация первой задачи. Изучены принципы работы голосового ассистента «Алиса» компании «Яндекс», интеллектуального помощника для смартфонов и персональных компьютеров, позволяющего решать задачи – поиск информации в Интернете, мест на карте, прокладывание маршрутов, сообщение прогноза погоды, поддержание разговора и развлечение пользователей. Для этого «Алиса» использует облачные средства компании «Яндекс» с обращением через API в Интернете.

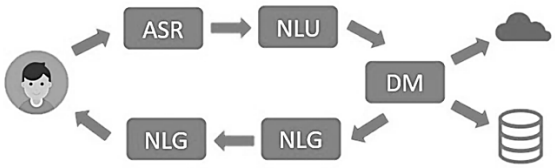

Рис. 1. *Принцип работы голосового ассистента «Алиса»*

На первом этапе происходит активация программы произношением ключевой фразы. Ассистент прослушивает окружающие звуки, анализирует наличие ключевой фразы и, когда она распознана, переходит в активный режим.

Пользователь произносит текст, который может объяснить помощнику, что пользователь хочет сделать. Система распознавания Automatic Speech Recognition (URL: https://usabilitygeek.com/automatic-speech-recognition-asr-software-an-introduction/
#:~:text=Automatic%20Speech%20Recognition%20or%20 ASR,variations%2C%20 resembles%20normal%20human%20conversation) превращает текст в N-лучших гипотез того, что сказал пользователь. Система распознавания естественного языка Natural Language Understanding (URL: https://www.techtarget.com/searchenterpriseai/
definition/natural-language-understanding-NLU#:~:text=Natural%20language%20 understanding%20is%20a, sentences%20using%20text%20or%20speech) превращает текст в N-лучших вариантов понимания фразы пользователя. Диалоговый движок интерпретирует и классифицирует эти фразы и определяет, что необходимо сделать на основе полученной информации, например, обратиться в сервисы для получения информации.

После получения необходимых данных система возвращает информацию пользователю, система генерации естественного языка Natural Language Generation (NLG. URL: https://www.techtarget.com/searchenterpriseai/definition/natural-language-generation
-
NLG#:~:text=Natural%20language%20generation%20(NLG)%20is,narratives%20from
%
20a%20data%20set) генерирует текст для ответа пользователю. Система генерации голоса Text-To-Speech (URL: https://cloud.google.com/text-to-speech) на основе обученных моделей генерирует звуковую информацию, которая объявляется пользователю в качестве ответной реакции. Кроме ответов могут происходить любые действия на мобильном телефоне или компьютере, например, запуск приложения или поиск информации в поисковой системе.

Недостатки голосовых ассистентов и актуальность развития. Голосовые ассистенты функционируют с 1916 г., но не получили широкого распространения из-за недостатков и ограничений сфер применения. Их недостатки – ориентированность на решения общих задач, зависимость от Интернета и облачных сервисов, сложность или невозможность интеграции со сторонними сервисами и незащищенность персональных данных.

Google Speech-To-Text API (Application Programming Interface. URL: https://cloud.google.com/speech-to-text) – описание способов взаимодействия одной компьютерной программы с другими, входит в описание интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы. Реализуется программной библиотекой или сервисом операционной системы. Используется программистами при написании приложений. С помощью этого набора компонентов программа, бот или сайт могут использовать другую программу. Google позволяет заимствовать или арендовать API за плату или

бесплатно. При желании использовать сервис можно зайти на сайт и протестировать его бесплатно. Корпорация Google разработала эффективный API для распознавания речи, который преобразует речь в текст – произносимый в микрофон текст в письменный текст в виде строк Python. Мы можем говорить в микрофон, и Google API переведет речь в письменный текст, работает с русским и английским языками. Мы можем отправить веб-запрос в API, который возвращает распознанный текст из кода Python, и скрипту (script) – сценарному языку потребуется доступ в Интернет.

Реализация второй задачи – установка программного обеспечения: Visual Studio Code. URL: https://code.visualstudio.com/; Python 3.11. URL: https://www.python.org/; Git. URL: https://git-scm.com/; ОС Windows 11. URL: https://support. microsoft.com/ru-ru/windows; Многоязычный синтезатор речи RHVoice. URL: https://github.com/RHVoice/ RHVoice.

Python 3.11 – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, на обеспечение переносимости написанных на нем программ. Язык Python 3.11 объектно-ориентированный с выделением блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, и почти нет необходимости обращаться к документации. Язык интерпретируемый и используется для написания скриптов.

Visual Studio Code – текстовый редактор, разработанный корпорацией Microsoft для Windows, Linux и macOS. Легкий редактор кода для кроссплатформенной разработки веб- и облачных приложений, включает отладчик, инструменты для работы с системой управления версиями Git, подсветку синтаксиса, технологию автодополнения IntelliSense (URL: https://learn.microsoft.com/ru-ru/visualstudio/ide/using-intellisense?
view=vs-2022) и средства для рефакторинга (refactoring) – перепроектирование кода. Имеет широкие возможности для настройки в соответствии с требованиями пользователей – пользовательские темы, сочетания клавиш и файлы конфигурации, распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

Система управления версиями **Git** создана Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена в 2005 г., последняя версия 2.40.0 с 13 марта 2023 г. (URL: https://lore.kernel.org/git/ 015d01d955ea$ 516c2390$f4446ab0$@nexbridge.com/T/#t).

ОС Windows – группа коммерческих проприетарных операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса. MS-DOS – предшественник Windows.

RHVoice – бесплатный синтезатор речи с открытым исходным кодом.

Для работы над проектом использовались библиотеки для языка Python.

PyAudio предоставляет кроссплатформенную библиотеку аудио ввода-вывода Python для PortAudio v19. С помощью PyAudio можно использовать Python для воспроизведения и записи аудио на платформах GNU/Linux, Microsoft Windows и Apple macOS.

Wikipedia API – простая в использовании Python-оболочка для API Википедии, поддерживает извлечение текстов, разделов, ссылок, категорий и переводов из Википедии. Документация содержит фрагменты кода для распространенных случаев использования.

google 3.0.0 – привязки Python к поисковой системе Google.

pyttsx3 – библиотека преобразования текста в речь на Python, в отличие от альтернативных библиотек, работает в автономном режиме и совместима с Python 2 и Python 3.

SpeechRecognition – библиотека для выполнения распознавания речи с поддержкой нескольких движков и API, онлайн и оффлайн.

DeepL Translator – гибкий бесплатный инструмент для перевода с разных языков с использованием нескольких переводчиков.

Реализация третьей задачи. Функционал ассистента. Базовый функционал – 1) распознавание речи пользователя и 2) синтез речи ассистента. Навыки: приветствие и рассказ о себе, вывод доступных команд на экран, генератор «Подбросить монетку», поиск информации в Google, видео на YouTube, в Википедии, перевод с английского языка на русский и обратно, открытие базовых программ.

Перед написанием и тестированием кода убедились, что операционная система готова к синтезу речи, в том числе на русском языке. Чтобы компьютер заговорил, нужны 1) голосовой движок – синтезатор речи с поддержкой нужных нам языков и 2) голоса дикторов для этого движка. В Windows есть штатный речевой интерфейс Microsoft Speech API (SAPI) 5.3 (URL: https://learn.microsoft.com/en-us/previous-versions/
windows/desktop/ms723627(v=vs.85)). Голоса к нему выпускают корпорации Microsoft, Nuance Communications (Nuance AI solutions. URL: https://www.nuance.com/index.html), Loquendo (Loquendo TTS. URL: https://only-soft.org/viewtopic.php?t=22383), компании Acapela Group (Acapela Voice Factory. URL: https://www.acapela-group.com/) и IVONA Software (IVONA Voices 2 для Windows. URL: https://nextup.com/ivona/).

Свободные кроссплатформенные голосовые движки. 1) eSpeak (URL: https://espeak.sourceforge.net/) и eSpeak NG (New Generation, новое поколение. URL: https://github.com/espeak-ng/espeak-ng) с поддержкой более 100 языков и диалектов, включая латынь. Система озвучит ваш текст в Windows, Android, Linux, Mac, BSD (Berkeley Software Distribution, распространялась Калифорнийским Университетом в г. Беркли. URL: https://www.bsd.org/). Система eSpeak стабильна в ОС Windows 7 и XP, а eSpeak NG совместима с Windows 8 и Windows 10.

2) Свободно распространяемый российский многоязычный синтезатор речи с открытым исходным кодом RHVoice от Ольги Яковлевой имеет четыре голоса для русского языка (мужской и 3 женских), поддерживает татарский, украинский, грузинский, киргизский, эсперанто и английский, работает в Windows, GNU/Linux и Android. Мы использовали RHVoice, и наш ассистент – кроссплатформенный.

Представляем часть кода проекта. Базовый функционал. Для начала импортировали **установленные библиотеки Python:** import speech_recognition # Распознавание пользовательской речи; import pyttsx3 # Синтез речи; import webbrowser # Работа с браузером; import traceback # Вывод traceback без остановки работы; import wikipediaapi # Найти в википедии; import random # Модуль рандомайзера; from deep_translator import GoogleTranslator # Модуль переводчика; import subprocess # Модуль запуска сторонних программ.

Затем необходимо **инициализировать синтез речи и найти на компьютере голосовые модули ассистента**, для этого используется **библиотека pyttsx3:**

```
“” “Инициализация синтеза речи” “”
tts = pyttsx3.init()
voices = tts.getProperty(“voices”)
# Задать голос по умолчанию
tts.setProperty(“voice”, “ru”)
# Попробовать установить предпочтительный голос
```

```
for voice in voices:
    ru = voice.id.find(“RHVoice\Anna”)  # Найти Анну от RHVoice
    if ru > -1:  # если нашли, выбираем этот голос
        tts.setProperty(“voice”, voice.id)
```

Для преобразования текста в речь написали функцию: def play_voice_assistant_speech(text_to_speech):

```
    “””
    :param text_to_speech: текст, который нужно преобразовать в
    речь
    “””
    tts.say(str(text_to_speech))
    tts.runAndWait()
```

Для преобразования речи пользователя в текст добавили функцию def record_and_recognize_audio(*args: tuple):

```
    “””

    Запись и распознавание аудио
    “””
    with microphone:
        recognized_data = “”
        # регулирование уровня окружающего шума
        recognizer.adjust_for_ambient_noise(microphone, duration=2)
        try:
            print(“Listening...”)
            audio = recognizer.listen(microphone, 10, 15)
        except speech_recognition.WaitTimeoutError:
            print(“Can you check if your microphone is on, please?”)
            return
        # использование online-распознавания через Google
        try:
            print(“Started recognition...”)
            recognized_data = recognizer.recognize_google(
                audio, language="ru"
            ).lower()
        except speech_recognition.UnknownValueError:
            pass
        “””
          в случае проблем с доступом в Интернет происходит выброс
        ошибки
                “””
        except speech_recognition.RequestError:
            print("Check your Internet Connection, please")
            play_voice_assistant_speech(
                “Пожалуйста, проверьте соединение с Интернетом!”
                )
        return recognized_data
```

Для обработки исключений здесь используется конструкция with – try – except.

Для запуска базовой программы написали main-функции if __name__ == "__main__": # инициализация инструментов распознавания и ввода речи.

Навыки ассистента. Сначала создали **словарь с кодовыми фразами и названиями функций:** commands = {(“подбрось”, “heads”): flip_a_coin, (“hello”, “hi”,

“morning”, “привет”, “здорова”, “хэй”): play_greetings, (“bye”, “goodbye”, “quit”, “exit”, “stop”, “пока”, “хватит”, “стоп”): play_farewell_and_quit, (“Victoria”, “help”, “вика”, “виктория”, “помощь”): name_trigger, (“search”, “google”, “find”, “найди”, “погода”, “прогноз”, “гугл”, “интернет”, “интернете”): search_for_term_on_google, (“video”, “youtube”, “watch”, “видео”, “ютуб”): search_for_video_on_youtube, (“Wikipedia”, “definition”, “about”, “определение”, “википедия”, “википедии”): search_for_definition_on_wikipedia, (“translate”, “interpretation”, “translation”, “перевод”, “перевести”, “переведи”, “переводчик”): get_translation, }.

Их **вывод для пользователя:** print (“Доступные команды:”, “Приветствие: Привет”, “Помощь (выводит это меню): Помощь; Виктория”, “Закончить разговор: Пока; Хватит; Стоп”, “Подбросить монетку: Подбрось монетку; Heads or tails”, “Запустить переводчик: Перевод; Перевести; Переведи;”, “Искать в Google: Найди; гугл; <запрос>”, “Искать в Википедии: Найди в википедии; <запрос>”, “Искать в Ютуб: ютуб; youtube <запрос>”, sep=”\n”)

Для выполнения этих команд добавили **функцию с дополнительными аргументами** def execute_command_with_name(command_name: str, *args: list).

Затем дописали **main-функцию с кодом для выполнения навыков ассистента** if __name__ == “__main__”: # инициализация инструментов распознавания и ввода речи.

Описание всех функций навыков. Поиск видео на YouTube: def search_for_video_on_youtube (*args: tuple)

Поиск в Google: def search_for_term_on_google (*args: tuple).

Поиск в Википедии: def search_for_definition_on_wikipedia (*args: tuple): Поиск в Wikipedia определения с последующим озвучиванием результатов и открытием ссылок.

Генератор «Подбросить монетку»: def flip_a_coin (*args: tuple): l = random.randint(0, 2).

Помощь: def name_trigger (*args: tuple): print (“Чем я могу помочь?”).

```
    play_voice_assistant_speech("Чем я могу пом+очь?”)
    print(
    “Доступные команды: “,
    “Приветствие: Привет”,
    “Помощь (выводит это меню): Помощь; Виктория”,
    “Закончить разговор: Пока; Хватит; Стоп”,
    “Подбросить монетку: Подбрось монетку; Heads or tails”,
    “Запустить переводчик: Перевод; Перевести; Переведи;”,
    “Искать в Google: Найди; гугл; <запрос>”,
    “Искать в Википедии: Найди в википедии; <запрос>”,
    “Искать в Ютуб: ютуб; youtube <запрос>”,
  sep=”\n”)
```

Переводчик: def get_translation (*args: tuple): “” “”; Переводчик; :param args:; :return:; “” “”;

```
    print(“Запускаю навык ‘Перевод’!”)
    play_voice_assistant_speech(“Запускаю навык ‘Перевод’!”)
    play_voice_assistant_speech(“Говорите целевой язык.”)
    lang = record_and_recognize_audio()
    if lang == “русский”:
        target = “ru”
    else:
        target = “en”
    play_voice_assistant_speech(“Говорите фразу для перевода.”)
```

```
        to_translate = record_and_recognize_audio()
        translated = GoogleTranslator(source='auto',
target=target).translate(to_translate)
        print(translated)
        play_voice_assistant_speech(translated)
        return
```

Поздороваться с пользователем: def play_greetings (*args: tuple): “”; Приветствие пользователя; “” “”;

```
        print (“Привет, пользователь! Я – голосовой помощник Виктория. Чем я могу помочь вам?”)
        play_voice_assistant_speech (“Привет, пользователь! Я – голосовой помощник Виктория. Чем я могу помочь вам?”)
```

Завершение программы: def play_farewell_and_quit (*args: tuple).

Открытие базовых программ: def open_app (*args: tuple): “”;

```
    Отрыть одну из стандартных программ Windows; “” “”;
    apps = {
        (“блокнот”, “notepad”): “notepad”,
        (“калькулятор”, “calculator”): “calc”,
        (“браузер”, “browser”): “browser”,
        (“wordpad”, “вордпад”): “write”,
        (“пэйнт”, “paint”): “mspaint”,
        (“проводник”, “explorer”): “explorer”
    }
    args = “join(args[0])
    error = 0
    for key in apps.keys():
        if args in key:
            if apps[key]!= “browser”:
                subprocess.Popen (apps[key])
                print (f’Запускаю {args}...’)
                play_voice_assistant_speech(f’Запускаю {args}...’)
                break
            else:
                webbrowser.open (‘https://google.com’)
                print (f’Запускаю {args}...’)
                play_voice_assistant_speech(f’Запускаю {args}...’)
                break
        else: error += 1
    if error >= len(apps):
        print (‘Такой программы не найдено...’)
        play_voice_assistant_speech (‘Такой программы не найдено.’)
```

Таким образом, в ходе выполнения проектных задач мы получили навыки написания работающей программы голосового ассистента.

Литература:

1. Дворянкин О.А. Голосовой помощник в интернете. Куда ведут нас информационные технологии// Молодой ученый. 2021. № 18 (360). С. 17-24.
2. Поляков Е.В., Мажанов М.С., Качалова М.В., Поляков С.В. Разработка интеллектуального голосового ассистента и исследование обучающей способности алгоритмов распознавания естественного языка [Электронный источник]// Системный администратор. 2017. № 12 (181). URL: https://samag.ru/archive/article/3570 (дата обращения: 15.04.2023).

3. Пишем голосового ассистента на Python. [Электронный источник] URL: https://habr.com/ru/post/529590/ (дата обращения: 15.04.2023).
4. Как сделать говорящую программу на Python самостоятельно? [Электронный источник] URL: https://gb.ru/posts/tts_python (дата обращения: 15.04.2023).
5. Распознавание речи в Python с использованием Google Speech API. [Электронный источник]. URL: https://progler.ru/blog/raspoznavanie-rechi-v-python-s-ispolzovaniem-google-speech-api (дата обращения: 15.04.2023).
6. Индекс пакетов Python (PyPI). [Электронный источник] URL: https://pypi.org (дата обращения: 15.04.2023).

Реквизиты публикации: *Деткова Л., Сухоручкина И., Севастьянов М. Цифровой голосовой помощник: проект по информатике и программированию// Учитель. 2023. № 2. С.13-20.*